M.S.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/539,197	03/30/2000	Kelly Gene Johnson	03048.P011	8640

23363 7590 11/06/2002

CHRISTIE, PARKER & HALE, LLP
350 WEST COLORADO BOULEVARD
SUITE 500
PASADENA, CA 91105

EXAMINER	
MAKHDOOM, SAMARINA	
ART UNIT	PAPER NUMBER
2123	

DATE MAILED: 11/06/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/539,197	Applicant(s) JOHNSON ET AL.	
	Examiner Samarina Makhdoom	Art Unit 2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 March 2000* .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) *2 & 5* .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

Specification

1. The specification is objected to because of the following informalities: The footer "0030487011" should be removed from all pages of the specification.

Appropriate correction is required.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) do not apply to the examination of this application as the application being examined was not (1) filed on or after November 29, 2000, or (2) voluntarily published under 35 U.S.C. 122(b). Therefore, this application is examined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

3. Claims 1-44 are rejected under 35 U.S.C. 102(e) as being anticipated by Hervin et al., U.S. Patent No. 6,205,560.

As per Claims 1, 17, 37-40, and 42-43, Hervin et al., disclose a method of debugging an executing service on a pipelined CPU architecture, the method comprising:

setting a breakpoint within an executing service (See Col. 4, lines 32-42 for setting and identifying breakpoints);

saving a minimum state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file);

altering a program counter of the executing service (See Col. 10, lines 15-37, for an alterable counter);

restoring the program counter of the executing service(See Col. 10, lines 15-37, for an alterable counter therefore it is able to go back to a saved value); and

restoring the state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file. A state written to a register file may be restored from the register file).

As per Claim 2, Hervin et al., disclose a executing debug commands within the executing service (See Abstract for executing debug commands for executing instructions).

As per Claims 3 and 19, Hervin et al., disclose locating an original instruction within the executing service to set the breakpoint (See Col. 2, lines 1-9 where Breakpoints are set at specific or original instructions or data in a program);

inserting a breakpoint instruction at the breakpoint (See Col. 2, lines 1-9 where breakpoints halt the program and shift the program's operation to the debug interrupt handlers i.e. the function of a breakpoint instruction);

starting the executing service (See Col. 2, lines 1-9 where breakpoints halt the program and shift the program's operation to the debug interrupt handlers i.e. start the execution);

waiting for the breakpoint to execute (See col. 4, lines 23-32 for the pipe control unit that keeps track of the earliest instructions, provides a stall output, and receives a delay input, i.e. waits for the breakpoint to execute);

waiting for memory fetches and configuration loads to complete (See Col. 6, lines 15-20 for the Memory/Bus controller that allows the breakpoint exception to complete); and

restoring the original instruction at the breakpoint location (See Col. 6, lines 1-3 where The write back WB stage updates the register file 24, condition codes, and other parts of the machine state with the results of the previously executed instruction therefore the processor has the information to restore the original instruction at the breakpoint location).

As per Claim 4, Hervin et al., disclose a altering an instruction within the executing service at a breakpoint location (See Col. 2, lines 46-50 for generating an exception that can go to many locations, therefore the instruction at the breakpoint is altered);

invalidating a page cache of the executing service (See Col. 5, lines 42-51 for cache address enablement, therefore, the cache can be invalidated).

As per Claims 5 and 21, Hervin et al., disclose a setting a breakpoint register to point to a breakpoint location (See Table 3, Definition of Debug Configuration Registers, Col 9 lines 10-45 for breakpoint registers).

As per Claim 6, Hervin et al., disclose saving the executing service registers (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file, since the register values determine the state, the register values are also saved);

and flushing a pipeline of the executing service (See Col. 4, lines 23-42 for the pipe control unit that flushed the pipeline).

As per Claim 7, Hervin et al., disclose determining if registers are unstable (See Figure 14 and text in Col. 18, lines 34-55, for the SMI interrupt for when the processor hangs, or the processor and registers are unstable);

if registers are unstable, saving the value of any registers that change after each pipeline cycle (See Col. 1, lines 19-29 for execution, the execution unit of the microprocessor uses a number of general purpose registers and control registers to write/read data or store control information. The data and/or control information at any point in time stored in the microprocessor, along with other information being used by the microprocessor, such as the instruction being fetched is called the processor state. Therefore, Hervin et al. disclose saving and restoring register values as part of saving and restoring the processor state); and

if the breakpoint location is set on a location that uses old values of registers, saving the old values of the registers before new values are written to the registers (See Col. 18, lines 34-55 for saving the processor state and system management mode so the program can return to that state or mode).

As per Claims 8 and 25, Hervin et al., disclose scalar registers or predicate registers (See Table 3, in Col. 9, lines 15-45 for the different types of registers and their functions).

As per Claims 9-10 and 26-27, Hervin et al., disclose setting the program counter of the executing service to point to a save stub (See Figure 5 and Col. 10, lines 10-37, for the break point program counter);

starting execution of the executing service (See Figure 6, and text in Col. 10, lines 38-67 for starting the execution);

executing the breakpoint (See Figure 6, and text in Col. 10, lines 38-67 for starting the execution, step 110 enabling the breakpoint);

storing configuration registers of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file, storing register values is part of storing the processors state);

saving values of scalar and predicate registers(See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file, storing register values is part of storing the processors state);

saving pipeline registers(See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file, storing register values is part of storing the processors state); and

storing a stack pointer value for a breakpoint location (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file, storing the stack pointer is part of storing the processors state).

As per Claims 11, 24, and 28, Hervin et al., disclose if a breakpoint location is on an instruction that does not make use of old values, restoring stable registers (See Col. 6, lines 40-58 for the breakpoint instruction does not make use of old values so a fault exception is generated);

if the breakpoint location is on an instruction that does make use of old values, restoring unstable registers, and reloading the pipeline(See Col. 6, lines 40-58 for the breakpoint instruction does not make use of old values so a trap exception is generated);

altering the program counter of the executing service to point to the breakpoint location; and starting execution of the executing service at the breakpoint location (Figure 3, and text in

Col. 6, lines 40-58 for the breakpoint instruction does not make use of old values so a trap exception is generated).

As per Claims 12, 16, 31-32, 36, 41, and 44, Hervin et al., disclose fetching a page of memory of the executing service into an instruction cache (See Col. 4, lines 49-56 for the instruction fetch stage);

checking for a checksum error within the page of memory (See Col 5, lines 1-18 for instruction decode stage that checks for errors in the instructions fetched from memory); and

if the executing service is set to reject the checksum error, saving the page of memory, inserting a breakpoint into the saved page of memory, altering an instruction pointer to the saved page of memory, and processing the saved page of memory (See Col. 8, lines 56-67 for an error in a group of instructions to set breakpoints and further debug the error).

As per Claims 13 and 33, Hervin et al., disclose a method of debugging an executing service on a pipelined CPU architecture, the method comprising:

setting a breakpoint at a last safe point (See Col. 4, lines 32-42 for setting and identifying breakpoints);

saving a minimum state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file);

simulating instructions of the executing service from the last safe point to the breakpoint (See Col. 8, lines 44-67 for regaining control of the program from the last breakpoint set);

executing debug commands within the executing service (See Col. 8, lines 44-67 for executing debug commands); and

restoring the state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file. A state written to a register file may be restored from the register file).

As per Claims 14 and 34, Hervin et al., disclose storing the simulated state of the executing server to the CPU (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file);

and restoring an original execution (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file. A state written to a register file may be restored from the register file).

As per Claims 15 and 35, Hervin et al., disclose comprises single stepping through a set of unsafe instructions, the set of unsafe instructions are between the last safe point and a next safe point (See Col. 8, lines 56-67 for the logical analyzer setting breakpoints in debug address register for all the instructions which helps to single step through a group of instructions).

As per Claim 18, Hervin et al., disclose a debugger to set a breakpoint within an executing service and execute debug commands within the executing service (See Abstract for a debugger for executing debug commands);

a save stub to save a minimum state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file) and alter a program counter of the executing service (See Col. 10, lines 15-37, for an alterable counter);

a processing engine to execute the breakpoint (See Abstract for a processing engine for debugging with breakpoints);

and a restore stub to restore the state of the executing service (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file. A state written to a register file may be restored from the register file).

As per Claim 20, Hervin et al., disclose a debugger is further operable to alter a page cache of the executing service (See Col. 4, lines 49-56 for fetching instruction and data into a cache, therefore the debugger can alter the cache).

As per Claim 22, Hervin et al., disclose saving the executing service registers (See Col. 6, lines 1-3 where the write back stage stores the processor state in a register file).

As per Claim 23, Hervin et al., disclose flushing a pipeline of a set of pipeline instructions of the executing service (Col. 4, lines 23-32 for the pipeline control unit for flushing the pipeline).

As per Claim 29, Hervin et al., disclose the restore stub is further operable to reload the pipeline state directly (See Col. 4, lines 28-42 for the pipe control unit to control the flow of the pipeline).

As per Claim 30, Hervin et al., disclose re-execute the original instructions within the pipeline to recreate the pipeline at a time of the breakpoint (See Col 7, line 60 to Col. 8, line 43 for the debug status register that reflects the conditions that were in effect at the time the breakpoint was enabled therefore this register reflects the original state of the pipeline).

Conclusion

4. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Deao et al., U.S. Patent No. 6,065,106 disclose debugging instructions in a processor and restoring the saved state.

Deao et al., U.S. Patent No. 6,112,298 disclose debugging instructions and controlling execution in a processor pipeline.

Groves, U.S. Patent No. 5,537, 536 discloses Apparatus and method for debugging electronic components through an in-circuit emulator.

Wood, T.J., "The test and debug features of the AMD-K7/sup TM/ microprocessor," Proceedings of the International Test Conference, 1999., Pages: 130 –136.

Vranken, H., "Debug facilities in the TriMedia CPU64 architecture," Proceedings of the European Test Workshop 1999., Pages: 76 –81.

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Samarina Makhdoom whose telephone number is 703-305-7209. The examiner can normally be reached on Full Time, Tuesday, Thursday, Friday, and Sunday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin J. Teska can be reached on 703-305-9704. The fax phone numbers for the organization where this application or proceeding is assigned are 703-305-0040 for regular communications and 703-305-0040 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900.

SM
November 1, 2002

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER